A networked control system as a case study for model checking of probabilistic hybrid systems *

Martin Fränzle, Tino Teige, and Andreas Eggers,
Carl von Ossietzky Universität Oldenburg, Germany
Dpt. of Computing Science, Research Group Hybrid Systems

September 4, 2009

1 Case study

In this benchmark description, we present a networked control system as an example for the quantitative analysis of probabilistic hybrid systems. Though the scenario of this case study is of academic nature, it is representative for a number of real-world industrial applications comprising the interplay of controlled continuous processes with feedback control mediated by a communication network exhibiting communication latencies. Among the frequently used communication protocols in such environments is the class of carrier sense multiple access protocols with collision detection (CSMA/CD) or with randomized collision avoidance (CSMA/CA), which are popular because of their relatively low average-case latency and the widespread availability of corresponding network components. The price to be paid is a broad variation in actual communication latencies stemming from retries after collision detection or random retreat in collision avoidance. The resulting large jitter in the end-to-end latency of the feedback path poses a major problem in networked control systems. A classical countermeasure is a pessimistic and thus costly dimensioning of the communication network, which results from both the aim of generally keeping the jitter low and the limited ability of current analysis methods to analyze such systems in their entity. The holistic model of the networked control loop presented herein, covers not only the effects of jitter on the controlled system, but also the otherwise often neglected dependency of communication frequency and thus collision probability and jitter on the state of the controlled system.

The case study is illustrated in Figure 1. Two processes are to be regulated by a networked control unit. These processes of potentially physical, biological,

*This work has been partially supported by the German Research Council (DFG) as part of the Transregional Collaborative Research Center "Automatic Verification and Analysis of Complex Systems" (SFB/TR 14 AVACS, `www.avacs.org`).

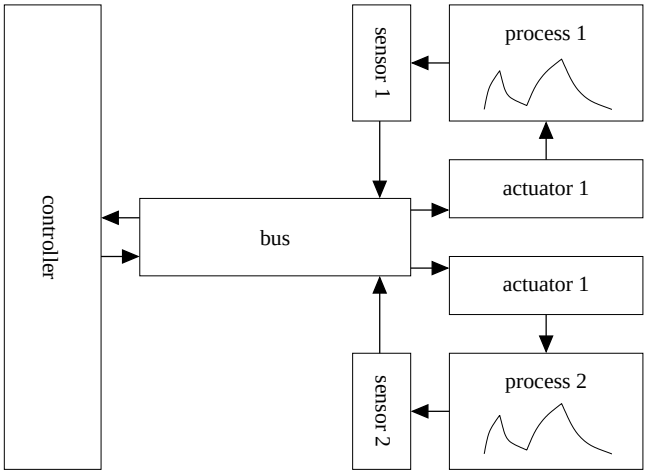

Figure 1: A networked controlled system

or chemical nature are continuously evolving while the actuators may occasionally switch some of their controlled inputs, causing a change of their continuous behaviors. Periodically, the observable states of the processes are measured by the sensors and then transmitted to the controller via the communication bus. The controller checks whether the current state of a process requires some modification of its inputs to change the process' behavior. If so, the controller sends an adequate data package over the bus to the corresponding actuator. In the present scenario we assume the bus to be a shared medium subject to collisions, i.e. only one of the sensors and controller may use the bus at a particular time. Obviously, this restriction leads to situations where a device is ready to send but cannot actually start sending due to bus occupancy, and also to situations where messages get lost due to simultaneous sending of more than one device. In both cases the protocol tries to resolve the undesired situation by random retreat, i.e., through assigning a random delay (before resend) to each of the conflicting senders. Given such a randomized protocol embedded in the described application, a quantitative analysis of the overall system requires a precise exploration of all the possible interplay between the continuously evolving processes, the timing delays of the discrete components, as well as the probabilistic choices in resolving potential communication conflicts.

2 Parallel components model of the case study

An implementation of the abstract case study motivated above is shown in Figure 2 and Figure 3. We have modelled the system by a set of parallel automata, each reflecting particular aspects of the system dynamics. Note that flattening out this concurrent system by a product construction would yield a PHA of $2^6 \cdot 3^2 \cdot 5^3 \cdot 7 = 504.000$ locations and, due to two further Boolean variables sig-

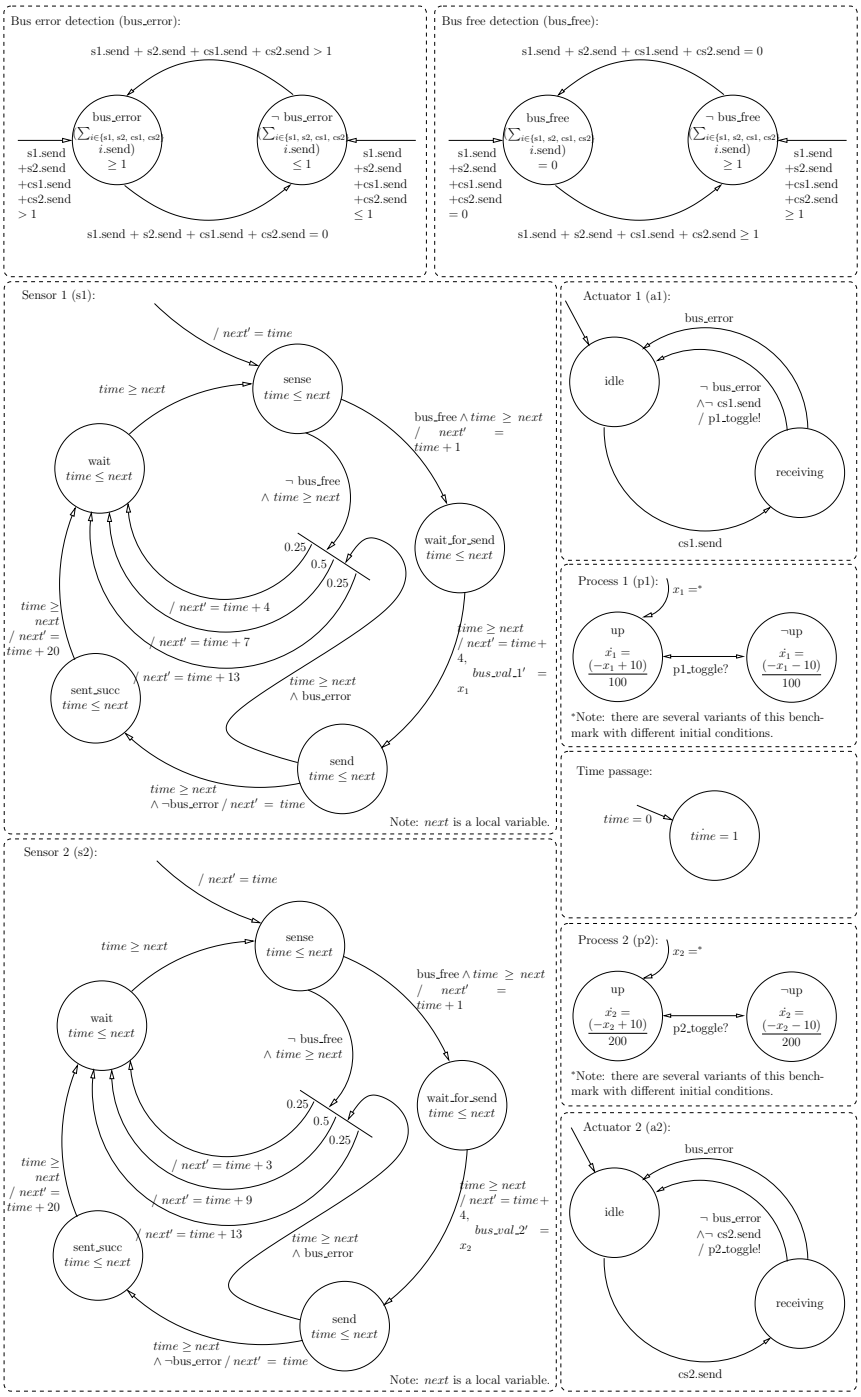

Figure 2: Parallel automata of the case study: bus, sensors, actuators, and continuous processes.

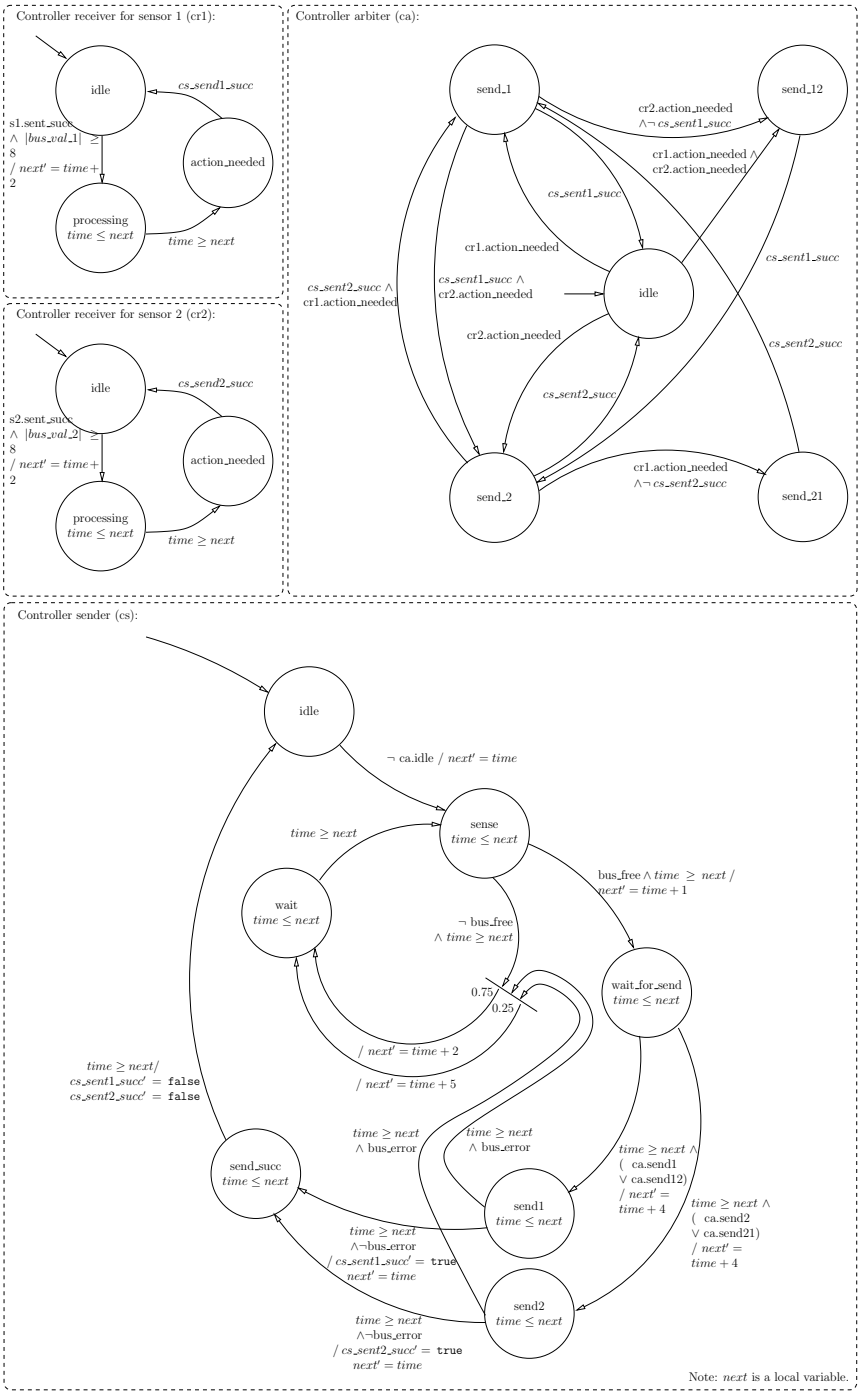

Figure 3: Parallel automata of the case study: parts of the controller.

nalling successful sends, slightly more than 2 million discrete states. In Figure 2, the plant model with its continuous evolution is given by two hybrid automata representing the two processes p1 and p2. If left uncontrolled, each of the processes converges exponentially to either +10 or -10, depending on the mode it is in. The task of the controller will be to keep the system inside the corridor $[-8.8, 8.8]$ by switching between the modes early enough. Therefore sensors, depicted by the automata s1 and s2, regularly try to send current measurements of the continous variables x_1 and x_2 representing the continous state of the plant over the bus to the controller. The controller (cf. Figure 3) comprises a receiver part for each of the sensors (cr1 and cr2), an arbiter (ca) to queue pending messages to be sent to the actuators, and a sender (cs) whose task is to transmit the actual messages to the actuators. In order to compensate for the anticipated network delay, the controller already decides to send a switch message, whenever a received sensor value lies outside the corridor of $[-8.0, 8.0]$. Successfully receiving such a switch message from the controller, the actuators (a1 and a2) perform the switch – modelled by sending an event p1_toggle or p2_toggle to the respective process p1 or p2.

As the bus protocol is one of the central aspects of the model, a more detailed explanation is of interest. As mentioned above, only one device can successfully transmit a message over the bus at a particular time. If more than one sender is using the bus, this can be detected afterwards – in practice by, e.g., bus snooping or through an acknowledge message from the adressed receiver or its absence. This behavior is modelled by the bus_error automaton, which switches to its error state as soon as there is more than one sender and leaves it only after the last of these senders has ceased sending. Thus, also the last sender can detect that its message has not been transmitted successfully.

According to the implemented protocol, a device has to sense the bus prior to actually starting to send in order to avoid unnecessary conflicts. If the bus is not free when sensing or an error occurred during transmission, a random retreat scheme is employed by setting a timer to a random value and waiting for the corresponding timespan. These random delays can be seen in the automata on the probabilistic transition from states sense to wait and send to wait respectively. For sensor s1, these are, e.g., 4 time units (tu) chosen with probability 0.25, 7 tu (probability 0.75), and 13 tu (probability 0.25). We chose lower delays for the controller because its messages have a higher urgency – it only sends a message if a toggle is really required. The sensor messages on the other hand will often contain measurements that will not cause any further reaction, as they will mostly frequently indicate that the current control mode can be held.

3 Encoding of the system

In the predicative encoding each instance of the variables describes the system's state at a particular point of time. Therefore, parallelism demands that each automaton can be interrupted and perform a transition at any moment – keeping

the valuation of its variables constant (often called *stutter jumps*). The formula thus explicitly encodes such self loops that are invisible in the automata depicted in the figures. The different forms of communication between the automata (using events, shared variables, or state observation) are all mapped to using the valuation of the corresponding variables in those parts of the formula encoding transition guards or actions. To compute the duration between two successive snapshots in a *scheduled event* fashion, we introduce a step variable for each of the components that is set to zero whenever the automaton switches between modes (which takes no time) or set to the flow duration when the automaton is known to reside in a mode (e.g. when a sensor is waiting). In the latter case, the *next* variables (encoding the temporal position of the next event) are used to determine these durations. The duration of the global step is thus the minimum of the local steps.

For more details, we refer the reader to the input files that contain the complete encoding.

We compare three different encodings of the continuous behavior. First, we encode the differential equation directly. As for this example, the closed-form solutions are easily computable, in a second encoding the ODE constraints are replaced by the explicit solution functions – thus forming a model without any ODE constraints. Assuming that such solution functions are not always easily obtainable, our third model comprises a safe overapproximation of the continuous behavior. In order to achieve this, we calculate the first Taylor terms of the ODE's exact solution and use the knowledge that any trajectory starting within $[-10, 10]$ will never leave that interval, to bound the error term. The resulting formula thus performs an Euler step and compensates for the truncation error using the bounded second order Taylor term. For the ODEs

$$\frac{d\,x}{d\,t} = \frac{-x(t) + 10}{r}, \text{ with } r \in \{100, 200\} \qquad \text{(I)}$$

$$\frac{d\,x}{d\,t} = \frac{-x(t) - 10}{r}, \text{ with } r \in \{100, 200\} \qquad \text{(II)}$$

the solution functions are given by

$$x(t_0 + h) = \quad 10 + e^{-h/r} \cdot (-10 + x(t_0)) \qquad \text{for (I)}$$

$$x(t_0 + h) = -10 + e^{-h/r} \cdot (\quad 10 + x(t_0)) \qquad \text{for (II)}$$

and the overapproximations by

$$x(t_0 + h) \in x(t_0) + h \cdot \frac{-x(t_0) + 10}{r} + \frac{h^2}{2} \cdot \frac{-[0, 20]}{r^2} \qquad \text{for (I)}$$

$$x(t_0 + h) \in x(t_0) + h \cdot \frac{-x(t_0) - 10}{r} + \frac{h^2}{2} \cdot \frac{-[-20, 0]}{r^2} \qquad \text{for (II)}$$

using $[-10, 10]$ as a safe overapproximation for $x(t)$ in the remainder term. Figure 4 illustrates the enclosure characteristics of this overapproximation and its conservativeness for different stepsizes h and $r = 100$.

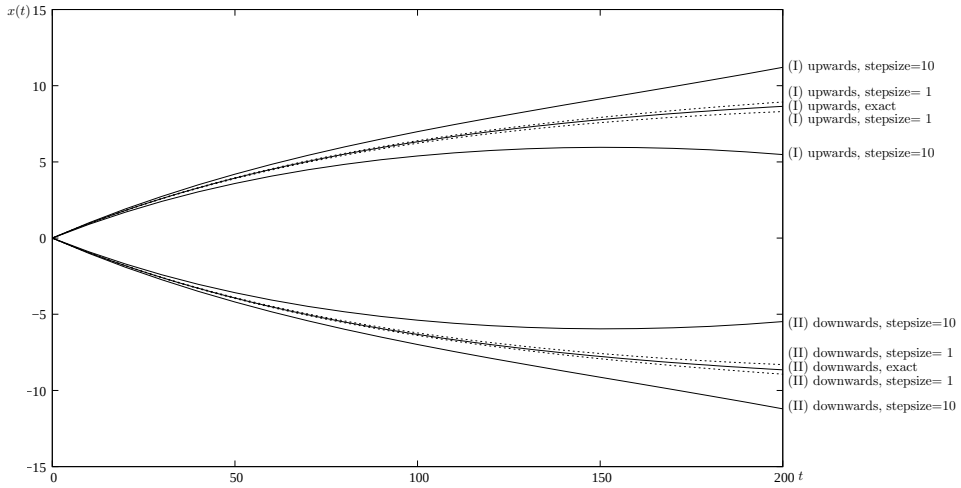

Figure 4: Overapproximation compared to exact solution.

Additionally, we define four scenarios, that have different characteristics regarding the amounts of probabilistic and existential quantification and non-determinism.

1. By replacing the random choices in the model by pre-defined constant values and setting the initial values of x_1 and x_2 to 0, the model becomes completely deterministic. We use this model to validate the behavior of the system.

2. Again fixing the initial values of x_1 and x_2 to 0 but leaving the randomized quantification intact, we get the smallest model that reflects the described behavior, i.e. faithfully represents collisions on the bus and the consequences of random retreat.

3. The third scenario also uses random quantification but adds existential quantification for the initial values. Instead of setting them to fixed values, both, x_1 and x_2 can be set to any value from $\{-2, -1, 0, 1, 2\}$ independently.

4. In the fourth scenario, a level of non-determinism for the initial values of $x1$ and x_2 is added. Instead of assigning the values from the existential quantifier's domain directly, they are used to map x_1 and x_2 to initial ranges from $\{[-2, -1], [-1, 0], [0, 1], [1, 2]\}$.

For each of these scenarios the three encodings have been generated. These can be found in the accompanying archive file.